APPENDIX G

Technology Demonstration System
Narrative Description

February 26, 1998

Contract No. 97-F131000-000

Delivered to the Office of Research and Development

2214 Rock Hill Road—Herndon, VA—20170

1.0 Introduction

This narrative describes the algorithms and techniques used by the Name Search - Technology Demonstration System (NS-TDS). It is the English-language version of the C++ source code that was used to develop the system. There are three major sections, covering NS-TDS support files, building the data base and performing a query. Each section relies on the contents of the previous section, so to effectively understand the system, this document should be read from beginning to end.

This narrative is tied to the source code through the use of paragraph numbers and comment lines. That is, whenever a block of code implements a technique or algorithm described in this document, a comment line has been inserted referencing the paragraph number. Comment lines are in the format, "// narrative paragraph number, x.x", where "x.x" stands for the paragraph number. If a block of code refers to more than one narrative paragraph, additional comments are added as separate lines.

2.0 Support Files

NS-TDS is a data-driven application dependent on a number of files that encapsulate years of computational linguistic research. These files represent the heart of the system and are essential to understanding how the primary algorithms work. This section of the narrative introduces these files by describing their purpose, contents and use.

2.1 Name Classifier Tables

TDS classifies the culture of a name as either Arabic, Chinese, Hispanic or "Other" (the default) by statistically analyzing its spelling. This analysis is accomplished with the aid of the following culture specific statistical distribution tables:

2.1.1 Digraph Score

Digraphs are contiguous letter pairs formed by parsing a name bracketed by a beginning and an ending boundary. For example, the name "FRED" consists of five digraphs: "#F", "FR", "RE", "ED" and "D#", where the symbol "#" represents a name boundary. In this table, digraphs that are clear indicators or contra-indicators of a particular culture are stored with a relative score. NS-TDS uses culture-specific tables to show the statistical likelihood of a particular digraph occurring in the applicable culture. For example, the digraph "QA" occurs almost exclusively in Arabic names, whereas the digraph "FM" almost never occurs in Arabic names. In the Arabic digraph table, "QA" is associated with a high positive score, and "FM" is associated with a low negative score.

2.1.2 Trigraph Score

Trigraphs are contiguous letter triplets that, for the purposes of TDS, are limited to the beginning and ending trigraphs. For example, the name "FRED" consists of the trigraphs "#FR" and "ED#". As with digraphs, NS-TDS uses culture specific tables to show the statistical likelihood of a particular trigraph occurring in the applicable culture.

2.1.3 Name Stop List

While generally good indicators of culture, digraph and trigraph distributions can erroneously classify specific names. For example, the name "BARKER" is identified as Arabic because it contains common Arabic letter patterns. The Name Stop List tables were implemented as a stopgap fix to this problem. For each culture, there is a Name Stop List table that contains a name along with a score that is either very positive (set to 2000) or zero (0). A high score means that name belongs to that culture; a score of zero means that the name does not belong there. So, "BARKER" is in the Arab Name Stop List with a score of 0.

The information in these tables is repeated for each culture and name part (i.e., given name or surname). For example, the following tables exist for Arabic:

agdi.dbf	Arabic digraph scores
agtri.dbf	Arabic trigraph scores
asdi.dbf	Arabic digraph scores
astri.dbf	Arabic trigraph scores

agnames.dbf	Arabic given name stop list
asnames.dbf	Arabic surname stop list

There are similarly named tables for Chinese starting with the letter “c” and tables for Hispanic starting with the letter “h”.

2.1.4 Phonetic Rules

In order to convert the spelling of a name into a phonetic representation, NS-TDS consults several rule files. They contain records that consist of search parameters based on spelling and replacement regular expressions based on International Phonetic Alphabet (IPA) characters. Take the following rule, for example:

> Boundary, “KN”, Vowel, “(kn|kan|n)

It says that if the letter string “KN” is found at the beginning of a name (“Boundary”) and is followed by a vowel, replace it with the IPA string, “(kn|kan|n)”, where “|” indicates “or”. The replacement string indicates that there are three possible pronunciations: [kn] or [kan] or [n]. (The use of square brackets is standard phonetic notation to indicate sounds rather than spelling). The spelling of a name is run through the rules until all characters are replaced with regular expressions. The name “KNOX” thus results in the regular expression (kn|kan|n)(a)(ks).

NS-TDS uses eight rule sets. For each of the four cultures (Anglo, Arabic, Chinese and Hispanic), there is a single vowel rule set and a multiple vowel rule set. The one vowel versions level all vowels to an [a] and produce fewer variations They are used for retrieval. The multiple vowel versions contain three basic vowel sounds, [a], [i] and [u], and are used in the ranking of retrieved names, since they are more precise than single-vowel rankings.

2.1.5 Simplified Phonetic Rules

One additional phonetic rule file is maintained to aid in the filtering process. It is a cross-reference file between all of the possible replacement strings in the single vowel rule sets and a simplified version of the replacement string. It is “simplified” in the sense that all *unbalanced* “ors” become *balanced.* For example, the replacement string (kn|kan|n) is “unbalanced” in that the possible pronunciations can contain one, two or three sounds. The simplified version is, (k?)(a?)(n), where “?” means that the sound is optional. The simplified string allows TDS to compare two regular expressions that may generate thousands of possible pronunciations with one calculation, thereby improving performance dramatically. Note that the simplified strings sometimes generate more possible pronunciations than the original replacement string, but never fewer. The additional pronunciations are handled adequately by the Ranker (see Section 4.5.1). Currently, this file is named *tds.simp_rul.*

2.1.6 Leveled IPA Matrix

Generating retrieval keys requires the creation of leveled IPA variant strings. That is, similar sounds (i.e., [s] and [z]) are treated as a single set. NS-TDS uses a cross reference file to define the set relationships. It is currently called, grouparray.dat.

2.1.7 Feature Difference Matrix

One of the key components of TDS is the ability to calculate a phonetic score when comparing two names. When comparing individual sounds, the calculation weights the difference between two sounds based on a *feature distance matrix*. This matrix consists of all combinations of two IPA characters and a score between 0.0 and 1.0 representing their phonetic proximity to one another, as defined by articulatory measures of similarity. It also contains records that represent the insertion or deletion of an extra sound. For example, the score assigned to the replacement of a [t] with a [d] is lower than the score assigned to the replacement of a [t] with a [k]. Further, inserting a vowel is given a lower score than inserting consonant such as [t] or [k].

The scores contained in this matrix reflect penalties. That is, higher scores mean that the sounds are further apart. All of the scores are based on linguistic principles of articulation, and reflect the number and type of phonetic features that cause the sounds to be different.

3.0 Building the NS-TDS Data Base

In order to search a large number of names quickly, NS-TDS uses a data base of name information and indices. This data base is built by a program hereafter referred to as the *Data Loader*. This program takes as input a text file of names that are preceded with a group ID. The group ID is a minor component of TDS that was implemented to facilitate an independent evaluation by ORD.

Building the data base consists of two major steps. First, the names are pre-processed to generate the information needed by the retrieval, filtering and ranking algorithms. This pre-processed data is stored in temporary tables that are subsequently turned into the NS-TDS data base and indices. The following paragraphs describe this process in detail. Where appropriate, examples are used to make the description easier to understand.

3.1 Pre-Process Names

All names are pre-processed to ensure validity (see 3.11) and to gather the information necessary for retrieval, filtering and ranking. This process shares many of the components used to pre-process a query name during an NS-TDS search.

3.1.1 Edit the Name

Input names are provided to the Data Loader in a text file and are edited according to the following specifications: Positions 1 through 6 must contain a group ID, where the first character must be a digit or the letter "Z". All other characters must contain a digit. Position 7 must be blank. Positions 8 though 37 contain the name and can only consist of upper case letters or an apostrophe. Furthermore, the name must be at least 2 characters in length and no longer than 30 characters. Any records that fail to follow the prescribed format are rejected and written to an error log, along with an appropriate message.

3.1.2 Classify the Name

The spelling of the name is statistically analyzed to determine the probable culture (Arabic, Chinese, Hispanic, or "Other"). This analysis is accomplished with the aid of the name classifier tables.

First, the name is parsed into digraphs (contiguous letter pairs) and beginning and ending trigraphs (contiguous three letter triplets) (see 2.1.1 and 2.1.2). Next, the digraphs and trigraphs are located in the appropriate classifier table to obtain the individual score. All of the scores are summed to obtain a total score. This process is repeated for all cultures.

Then the Name Stop List tables for each culture are checked. If the name is found in one of the tables, the associated score is returned ("2000" means in the culture, "0" means not in the culture). If the name is found, the previously calculated culture score is replaced.

Finally, each score is compared to a culture-specific threshold. If no scores exceed the culture threshold, the name is classified as "Other". If one score exceeds the appropriate threshold, the name is classified accordingly. If more than one score exceeds the culture threshold, the highest score is chosen and that culture is returned. If there is a tie (very unlikely), the culture is chosen alphabetically with Arabic first followed by Chinese and then Hispanic. It is important to note that an input name will receive only one classification.

3.1.3 Generate 1 Vowel Regular Expressions

In this step, the spelling of the name is run through the spelling-to-IPA phonetic conversion rules, to generate a regular expression that represents all of the possible pronunciations of the name. Every name is run through the single-vowel Anglo phonetic rule set, which is the default/generic rule set. If the name was classified as Arabic, Chinese or Hispanic, it is also run through the appropriate single-vowel rule set for that culture, generating a second IPA regular expression.

3.1.4 Generate Simplified Regular Expressions

Using the simplified phonetic rules, an simplified regular expression is generated. The expression is encoded into compact byte representations to make further calculations faster. As before, if the name was classified as Arabic, Chinese or Hispanic, a second simplified regular expression is generated according to the appropriate rule set.

3.1.5 Generate 1 Vowel Variants

Using the generated regular expression, a list of possible IPA variants is generated and added to a temporary table of variants for all input names. As an example, the name "KNOX", which generates the regular expression (kn|kan|n)(a)(ks), generates the following variants: [knaks], [kanaks], and [naks]. The temporary table lists all variants, as well as the name that generated the variant. It used later in the data base build process.

3.1.6 Determine the Initial Consonants

The variants are then analyzed, to generate a list of all possible name-initial IPA consonants. For example, the name "KNOX" starts with the regular expression (kn|kan|n), which can have an initial consonant of [k] or [n]. Note that if the variant starts with an IPA vowel, the first IPA consonant is used to build this list. Thus the name O'NEIL would have [n] as the initial consonant. This information is used by the Ranker (see section 4.4.4 below).

3.1.7 Set the Initial Vowel Switch

Next, the variants are analyzed to determine if it is possible for the pronunciation of the name to start with a vowel. It is a three-way switch that indicates whether the pronunciation (1) can never start with a vowel, (2) can sometimes starts with a vowel or (3) always starts with a vowel. This information is used by the Ranker (see section 4.4.5 below).

3.2 Build Data Base and Indices

This step takes all of the information produced during pre-processing and builds the data base and indices used by TDS for retrieval, filtering and ranking.

3.2.1 Create Name Files

A name file is generated for all four cultures processed by TDS (Anglo, Arabic, Chinese and Hispanic). "Anglo" represents the default, and therefore all names generate an Anglo record; only those names that are appropriately classified generate records in the other culture name files. Each record in the name file contains: the spelling of the name, the simplified regular expression codes, the list of initial consonants, the initial vowel switch, the group ID and an internal unique ID.

The naming convention is a four-letter culture identification followed by the extension, "nam". Currently, the following name files are generated: *angl.nam*, *arab.nam*, *chin.nam* and *hisp.nam*.

3.2.2 Generate Leveled Variants

Using the variants generated during pre-processing and the leveled IPA matrix, a list of leveled variants is built. Furthermore, the input variants have duplicate contiguous characters removed. The IPA characters in "KNOX" generate the following numeric codes, based on sets of similar sounds: [k] = 5; [n] = 2; [a] = 0; [s] = 4; [z] = 4. (Note that [s] and [z] are both indexed as "4", since they are similar sounds). The following unique leveled variants are generated: 52054, 502054 and 2054. Note that the number of leveled variants is usually less than the number of non-leveled variants. For each input name, the leveled variants are added to a temporary file that lists all leveled variants and the name that generated it.

3.2.3 Create Retrieval Indices

Retrieval indices consist of a unique sorted list of leveled variants. As with the name files, one index is generated for each culture. Each index is created by sorting and then *deduping*, i.e., removing duplicate forms from the previously-built temporary file of leveled variants.

The files produced by this step are named *angl.idx*, *arab.idx*, *chin.idx* and *hisp.idx*.

3.2.4 Create Index-to-Name Maps

Finally, a map file is created that cross-references all of the index records with the name records that generated the leveled variant. The retrieval index records contain a pointer to a map record. The map record contains a list of pointers to name records. This structure allows TDS to quickly scan the indices and generate a list of candidate names during retrieval.

The names of the map files are *angl.vec*, *arab.vec*, *chin.vec*, and *hisp.vec*.

4.0 Performing a Query

The heart of NS-TDS is the ability to perform a query that returns a ranked list of results. This is done using five major steps: query pre-processing, exact phonetic search, similar phonetic search, initial ranking and final ranking. The following paragraphs describe these steps and their components in some detail. When appropriate, examples are used to make the descriptions easier to understand.

4.1 Pre-Process Names

The query name is pre-processed to ensure validity and to gather the information necessary for retrieval, filtering and ranking. This process shares many of the components used to pre-process an input name during the building of the NS-TDS data base.

4.1.1 Edit Name

After the user enters a query name via the user interface, it is edited according to the following criteria: The name can only consist of the 26 letters of the Roman alphabet or an apostrophe. Further, the name must be at least 2 characters in length and no longer than 30 characters. Errors are displayed to the user in a dialog box, along with an appropriate message.

4.1.2 Classify Name

If the user has specified that culture classification is automatic (the default), the spelling of the name is statistically analyzed to determine the probabie culture (Arabic, Chinese, Hispanic, or "Other"). This analysis is accomplished with the aid of the previously described name classifier tables (see 2.1 above). If the user has overridden the default and manually specified a culture, this step is skipped.

The name is then parsed into digraphs (contiguous letter pairs) and beginning and ending trigraphs (contiguous three letter triplets). Then, the digraphs and trigraphs are located in the appropriate classifier table to obtain the individual score. All of the scores are summed to obtain a total score. This process is repeated for all cultures.

Next, the Name Stop List tables (see 2.1.3 above) for each culture are checked. If the name is found in one of the tables the associated score is returned ("2000" means in the culture, "0" means not in the culture). If the name is found, the previously calculated culture score is replaced.

Finally, each score is compared to a culture specific threshold. If no scores exceed the culture threshold, the name is classified as "Other". If one score exceeds the appropriate threshold, the name is classified accordingly. If more than one score exceeds the culture threshold, the highest score is chosen and that culture is returned. If there is a tie (very unlikely), the culture is chosen alphabetically, with Arabic first, followed by Chinese and then Hispanic.

It is important to note that an input name will receive only one classification. Further, if the classification is Arabic, Chinese or Hispanic, all further pre-processing will be performed twice (once for the default Anglo culture and once for the culture identified by classification or as manually specified by the user).

4.1.3 Generate 3 Vowel Regular Expressions

In this step, the spelling of the name is run through the multiple vowel phonetic rules to generate a state table that represents all of the possible pronunciations of the name in IPA form. Every name is run through the default Anglo phonetic ruie set. If the name was classified as Arabic, Chinese or Hispanic, it is also run through the appropriate rule set for that culture generating a second state table. These will be used during the *exact* phonetic search (see 4.2).

4.1.4 Generate Multiple (3) Vowel Variants

Using the multiple vowel state table, a list of all possible IPA variants is generated. As an example, the name, "KNOX", which generates the regular expression (kn|kan|n)(a|u)(ks), generates the following IPA variants: [naks], [nuks], [kanaks], [kanuks], [knaks], and [knuks]. This list will be used to perform a brute force phonetic score adjustment on names that pass preliminary ranking (see 4.5).

4.1.5 Generate 1 Vowel Variants

Using the 1-vowel state table, a list of all possible one-vowel IPA variants is generated. As an example, the name, "KNOX", which generates the regular expression, (kn|kan|n)(a)(ks), generates the following variants: [naks], [kanaks], and [knaks]. This list will be used to generate retrieval and ranking information.

4.1.6 Determine the Initial Consonants

The single-vowel variants are then analyzed to generate a list of all possible initial IPA consonants. For example, the name KNOX starts with the regular expression, (kn|kan|n), which can have an initial consonant of [k] or [n]. Note that if the name starts with a vowel, the first IPA consonant is used to build this list. Thus, the name, O'NEIL would have [n] as the initial consonant. This information is used during ranking (see 4.4.4).

4.1.7 Set the Initial Vowel Switch

Next, the single-vowel variants are analyzed to determine if it is possible for the pronunciation of the name to start with a vowel. It is a three-way switch that indicates that the pronunciation (1) can never start with a vowel, (2) can sometimes start with a vowel, or (3) always starts with a vowel. This information is used during ranking (see 4.4.5).

4.1.8 Generate Leveled Variants

Using the appropriate one-vowel rule set, a temporary list of all possible IPA variants is generated. This list, along with the leveled IPA matrix, is used to build a list of leveled variants. Also note that duplicate contiguous characters are removed. For example, the IPA characters in KNOX generate the following numeric codes, based on sets of similar sounds: [k] = 5; [n] = 2; [a] = 0; [s] = 4; [z] = 4. (Note that [s] and [z] are both indexed as "4", since they are similar sounds). The following unique leveled variants are generated: 52054, 502054 and 2054. Note that the number of leveled variants is usually less than the number of non-leveled variants.

4.1.9 Generate Simplified Regular Expressions

Using the simplified phonetic rules, a simplified regular expression is generated. The expression is encoded into compact byte representations to make further calculations faster. As before, if the name was classified as Arabic, Chinese or Hispanic, a second simplified regular expression is also generated.

4.1.10 Initialize Search Parameters

The search parameters specified by the user or defaulted by the application are stored along with the query information. These parameters set thresholds for retrieval and filtering, and determine the weights given to individual ranking scores.

4.2 Exact Phonetic Match

An exact phonetic search is always performed by TDS. It is a quick search that retrieves names which share at least one possible pronunciation with the query name and passes them to the ranker. A search of the Anglo data base is always performed; if a non-Anglo culture was determined or specified by the user, the search is repeated for the appropriate culture.

4.2.1 Retrieve Candidates

Each of the leveled variants generated by pre-processing the query name are used as a key to perform a binary search of the retrieval index, which is a set of unique leveled variants for the name data base. In the case of "KNOX", three leveled variant indices are retrieved: 2054, 502054 and 52054.

4.2.2 Retrieve Name Information

Using the index-to-name map files, all data base names and associated information that could possibly generate the leveled variants found above are put into a list. In the case of "KNOX", names such as "NOCKS", "NOX", "KNOCKS" and "NAUCHS" are returned.

4.2.3 Execute Exact Phonetic Match Algorithm

For each name retrieved, a regular expression is generated using the appropriate multiple-vowel rule set. Each of these is compared to the query's regular expression to determine if there is an intersection. In other words, the two expressions are evaluated to see if they can generate a matching variant. This evaluation is done by generating non-deterministic finite state tables and walking through each table until a match is impossible (i.e., the names do not match), or the end of both tables is reached (i.e., the names match). If the name passes this algorithm, it is placed in a list.

4.2.4 Pass Exact Matches to the Ranker

All names that pass the exact-match algorithm are sent to the Ranker, along with the information retrieved from the name file. In addition, the phonetic score is set to 1.0, which is the highest possible score, and the pipe (rule set) that was used to retrieve the name is passed. Note that if a name was found to be an exact match under two cultures, it is included twice.

4.3 Similar Phonetic Match

A similar phonetic search is performed only if the user has requested it. Note that the default is to perform a similar search. It is slower and more thorough than the exact search, and retrieves names that sound similar (based on principles of articulation) to the query name to the ranker,. A search of the Anglo data base is always performed; if a non-Anglo culture was determined or specified by the user, the search is repeated for the appropriate culture.

4.3.1 Scan the Retrieval Index

A complete scan of the retrieval index is performed, and each leveled variant is compared to the leveled variants generated by pre-processing the query. The comparison uses a standard edit distance calculation to determine how far apart two strings are. The algorithm determines the minimum number of edits (insertion, deletion or replacement of IPA characters) necessary to convert one string into another. A score is calculated by dividing the number of edits by the maximum length of the two strings and subtracting this fraction from 1 resulting in a score between 0.0 and 1.0. This score is compared to the retriever threshold, and those records with a score greater than or equal to the threshold are added to a candidate list.

4.3.2 Filter the Candidates List

This list is scanned and, if the name has not already been retrieved by the exact match algorithm, the simplified regular expression of the query name is compared to simplified regular expressions of all of the candidate names. This comparison uses a more linguistically sophisticated edit distance algorithm that takes into account the phonetic features of each sound. All edits are weighted according to the relationships stored in the Feature Distance Matrix, For example, the replacement of similar sounds that share most phonetic characteristics, like [s] and [z], are given a small penalty. The "cost" of replacements is determined by where and how sounds are articulated in the mouth and the "effort" required to produce one rather than the other. Similarly, some insertions and deletions of sounds are more costly than others (e.g., insertion of a [t] is more costly than insertion of a vowel, [a]). So, instead of computing the minimum number of edits required to convert one string into another, this algorithm calculates the path of least resistance. As with the retrieval calculation, a score between 0.0 and 1.0 is obtained by dividing the total penalty by the maximum length and subtracting this fraction from 1. This score is compared to the filter threshold, and those records whose score is less than the threshold are discarded.

Finally, it is important to note that because simplified regular expressions can generate more variants than the expressions they were derived from, it is possible for this score to be higher than expected, although it is impossible to obtain a lower score. This deficiency is corrected during ranking.

4.3.3 Pass Similar Matches to the Ranker

For all records that pass the filter algorithm, additional information is gathered from the name file via the index-to-name map. Also, the phonetic score of these names is set to the score calculated during the filter edit distance calculation, and the culture pipe (i.e., rule set) that was used to retrieve the name is passed. This list is sent to the Ranker for initial scoring. Note that if a name was found to be a similar match under two cultures, it is included twice.

4.4 Initial Ranking

Names that pass the retrieval and filter stages via the exact or similar phonetic match searches are sent to the Ranker, along with the phonetic score calculated by the filter and all of the data built during the pre-processing stages (initial consonant, initial vowel, etc.). The ranker also knows which search (exact or similar) produced the return. It takes this information, calculates several other scores, applies weights to those scores based on the query parameters and produces a ranked list of names with combined scores. Initial ranking differs from final ranking in that it uses the phonetic score calculated by the filter. Final ranking, which is described below (see 4.5), performs a more exhaustive and exact phonetic score calculation.

4.4.1 Calculate the Spell 1 Score

Because spelling is a relevant factor in determining similarity of names, the Ranker is set up to consider spelling in its calculations in ranking of names passed to it. In the case of exact matches, for example, all phonetic scores are 1.0, but spellings can vary widely (e.g., “LI”, “LEE”, “LEIGH”). The *Spell 1* score is a comparison of all the letters in the query name to all of the letters in the data base name. Each letter that matches contributes to the score, and no letter can be used more than once. Note that the position of the letter has no bearing of the score. So, the “K” in “KNOX” matches the “K” in “SACK”. In addition, there is an option to bias the score so that letters on the left side of the query name count more than those towards the end. The “left-bias” factor defaults to true. A score is calculated by dividing the value of the matches (an integer, if left bias is not used) by the maximum length of the query and data base name and then subtracting this fraction from 1.0. This results in a score between 0.0 and 1.0.

4.4.2 Calculate the Spell 2 Score

The *Spell 2* score works similarly to *Spell 1*, except that it uses *digraphs* instead of single letters. Digraphs are contiguous letter pairs formed by parsing a name bracketed by a beginning and an ending boundary. For example, the name “FRED” consists of five Roman character digraphs: “#F”, “FR”, “RE”, “ED” and “D#”, where “#” represents a name boundary. Digraphs build some contextual information into the calculation, with the result that “FRED” and “BRID”, which share two non-contiguous letters, have a lower Spell 2 score than a Spell 1 score. Spell 2 uses the same left bias parameter and the same method to turn the calculation into a decimal number between 0.0 and 1.0 as Spell 1. Finally, the Spell 2 score contains a special adjustment for names that start with a vowel. For example, when comparing the name, “NEIL” to “ONEIL” and “SNEIL”, the score for “NEIL” will be adjusted upwards by a small factor.

4.4.3 Calculate the Syllable Score

The Syllable score compares the number of syllables in the query name to the number of syllables in the data base name. Counting the number of syllables in a name is based on the spelling, and essentially says that a syllable occurs when there are one or more vowels in a row, preceded by a consonant or a word boundary. Adjustments are made for special cases such as, dipthongs (multiple consecutive vowels pronounced as two syllables) and “E” or “ES” at the end of name which often does not produce a separate syllable. A score is produced by dividing the difference in the number of syllables between the query name and the data base name by the maximum number of syllables in the query or data base name and subtracting the resulting fraction from 1.0. This results in a score between 0.0 and 1.0.

4.4.4 Calculate the Initial Consonant Score

The initial consonant sound in names hold particularly prominent positions in determining similarity of names. The Initial Consonant score compares the first occurrence of an IPA consonant in the query name to that of the data base name. The consonants are compared based on the Feature Distance Matrix, producing a score between 0.0 and 1.0. For example, [s] and [z] will return a high score, whereas, [k] and [r] will return a low score. Note that the first consonant can be different as is the case with the name, "KNOPF", which could start with a [k] sound or an [n] sound. The algorithm compares all possibilities and returns the best possible score.

4.4.5 Calculate the Initial Vowel Score

The Initial Vowel score comes into play when both the query name and the data base name start with a vowel. If this is not the case, the initial vowel score is 1.0. Otherwise, the IPA vowel or vowels that start the names are compared and a score is returned based on the feature distances between them. The score is a decimal between 0.0 and 1.0.

4.4.6 Calculate the Culture Score

The culture score compares the culture of the query name as determined by the classifier or as specified by the user with the "pipe"(rule set) used to retrieve the data base name. So, if a name is classified as Arabic, and the name being ranked was passed to the ranker via the Arabic pipe, the culture score is 1.0. If the query culture does not match the pipe used to retrieve the name, the culture score is 0.0. This allows the Ranker to "bump up" names that share the same cultural identity, as in Chinese "CHIN" and "CHANG" (versus non-Chinese "CHAIN", for example).

4.4.7 Calculate the Final Score

The final or total score is an amalgamation of all of all the previous scores. NS-TDS maintains a set of parameters that allows the user to assign weights to each of the various individual scores. Note that in the user version, these weights are not modifiable. The weights are intended to be percentages, so that each factor is a decimal between 0.0 and 1.0 and the total adds up to 1.0. This is not a requirement, as the calculation recomputes the weights relative to one another.

So, to arrive at the final score, all of the individual scores are multiplied by their weight and the results are summed. This results in a decimal score between 0.0 and 1.0 with a higher score indicative of a better match.

4.4.8 Set the Ranking Order

The absolute ranking of data base names is based on whether or not the name is an exact phonetic match and on the value of the final score. Exact phonetic matches are always ranked first, followed by similar matches. Within these two categories, names are ranked according to the final score, with higher scores ranked at the top of the list. It is quite possible that an exact match will receive a lower final score than a similar match (if its spelling and/or culture scores are low, for example), and yet be ranked above the similar match based on its category of "exact match". For example, "KNOX" returns the exact match "NAX" with a lower score (.825) than the similar match "KNAGGS" (.848), but forces *all* exact matches, including "NAX", to the top of the list.

4.4.9 Return a Ranked Set

Finally, the Ranker eliminates names that do not meet or exceed the threshold set in the NS-TDS parameters, unless the name is considered an exact match. Exact phonetic matches are always returned, regardless of their score.

4.5 Final Ranking

The purpose of Final Ranking is to incorporate a more accurate phonetic score into the overall ranking. Recall that the phonetic score used by the initial Ranker is calculated by the Filter, using simple regular expressions. This algorithm, while fast, can inflate the phonetic score, producing inaccurate ranking. Further, the Filter calculates using single vowel rules, which can introduce another source of inaccuracy (e.g., "LITZ" = "LUTZ"). Final ranking adjusts the phonetic score by performing a brute force edit distance calculation, using multiple vowel rules. This calculation is performed at this point because it is time-consuming, and must be limited to the smallest possible set of input data to meet performance requirements. Note that Final Ranking is a parameter option, although the default is set to true.

4.5.1 Recalculate the Phonetic Score

All names that were retrieved via the similar phonetic search and passed initial ranking are reprocessed to produce an accurate phonetic score. Names that were retrieved via the exact phonetic match search do not need to be recalculated because their phonetic score is always 1.0. First, the names are passed through the appropriate cultural multiple vowel rule set to produce a list of all possible IPA variants. Then, a brute force edit calculation is performed; every variant from the query name is compared to every variant of the data base name by performing a phonetic edit distance calculation. The best score is retained and assigned to the result.

4.5.2 Recalculate the Final Score

Using the same logic as that used by the initial Ranker, the Final Score is calculated using the new, more precise phonetic score. This will result in lower scores for some names; if these names fall below the final score threshold, they are removed from the ranked list.

4.5.3 Rebuild the Ranked Set

Finally, using the new score, the set of names is ranked again; with exact matches at the top followed by similar matches. Also, the final ranker will only return the maximum number of names requested by the user. The default setting is 145. Thus, it is possible for a name to pass NS-TDS, but not be displayed on output.